# List of 6th sem IOT/Embedded system programs

**1. Programs to use GPIO with LED / Buzzer with interrupt int1/int0**
**//when interrupted with B11 buzzer beeps (connect to gnd)**
**//when interrupted with C15 led glows (connect gnd)**

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvSYS.h"

void Led_CALLBACK (void)
{
        DrvGPIO_ClrBit(E_GPC, 15); // output Low to turn on LED
        DrvSYS_Delay(300000);    // delay
        DrvGPIO_SetBit(E_GPC, 15); // output Hi to turn off LED
        DrvSYS_Delay(300000);    // delay
}

void Buzzer_Callback(void)
{
        DrvGPIO_ClrBit(E_GPB,11); // GPB11 = 0 to turn on Buzzer
        DrvSYS_Delay(100000);            // Delay
        DrvGPIO_SetBit(E_GPB,11); // GPB11 = 1 to turn off Buzzer
        DrvSYS_Delay(100000);            // Delay
}

int main (void)
{
        UNLOCKREG();                                    // unlock register for programming
        DrvSYS_Open(48000000);// set System Clock to run at 48MHz     // 12MHz crystal input, PLL
output 48MHz
        LOCKREG();
        // lock register from programming

        // Initialize LEDs (four on-board LEDs below LCD panel)
        DrvGPIO_Open(E_GPC, 15, E_IO_OUTPUT); // GPC12 pin set to output mode
        DrvGPIO_SetBit(E_GPC, 15);                // Goutput Hi to turn off LED

        //DrvGPIO_Open(E_GPB, 11, E_IO_INPUT);
```

```
        DrvGPIO_EnableEINT0(E_IO_RISING, E_MODE_EDGE,Led_CALLBACK); //GPIO port E_GPB, pin 14
        DrvGPIO_Open(E_GPB, 11, E_IO_OUTPUT); // initial GPIO pin GPB11 for controlling Buzzer

        // External Interrupt
        DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);            // configure external interrupt pin GPB15
        DrvGPIO_EnableEINT1(E_IO_BOTH_EDGE, E_MODE_EDGE, Buzzer_Callback);
         // configure external interrupt

        while(1)
        {
        }

}
```

**2. Program to use GPIO as input from A port and display the port bit number**
**//shows port::f with blue led light**

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"
#include "LCD_Driver.h"

int main()
{
char TEXT[16];
int a;

UNLOCKREG();
SYSCLK->PWRCON.XTL12M_EN=1;
DrvSYS_Delay(5000); // Waiting for 12M Xtal stalble
SYSCLK->CLKSEL0.HCLK_S=0;
LOCKREG();

DrvGPIO_SetPortBits(E_GPA,15);
a=DrvGPIO_GetPortBits(E_GPA);

Initial_panel();
clr_all_panel();

//to print decimal: sprintf(TEXT,"port is %d",a);
sprintf(TEXT,"port :: %x",a);
```

```
print_lcd(0, TEXT);
}
```

**3. Program interrupt with port A and identify A port bit that was interrupted and increment the counter to count no of interrupts**

```
//interrupt with GPA15 to GND counter increments
//displays port::fff
/ /IRQ_A : 2
// ^increments
//GPA interrupt!

#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"
#include "LCD_Driver.h"
int counter = 0;
//USER DEFINED FUNCTION
void GPIO_GPAB_CallBack()
{
int a;
char TEXT1[16];
//Highlighted one not needed
counter++;
print_lcd(3,"GPA interrupt !!");
a=DrvGPIO_GetPortBits(E_GPA);
sprintf(TEXT1,"port :: %x",a);
print_lcd(0, TEXT1);
}

int main()
{
char TEXT[16];

UNLOCKREG();
SYSCLK->PWRCON.XTL12M_EN=1;
DrvSYS_Delay(5000);
SYSCLK->CLKSEL0.HCLK_S=0;
```

```
LOCKREG();

// setup GPA15 to get interrupt input
DrvGPIO_Open(E_GPA,15,E_IO_INPUT);
DrvGPIO_EnableInt(E_GPA, 15, E_IO_RISING, E_MODE_EDGE);
DrvGPIO_SetDebounceTime(5, 1);
DrvGPIO_EnableDebounce(E_GPA, 15);
DrvGPIO_SetIntCallback(GPIO_GPAB_CallBack,NULL);

Initial_panel();
clr_all_panel();

while(1)
{
sprintf(TEXT,"IRQ_A: %d",counter);
print_lcd(1, TEXT);
}

}
```

**4. Program for using ADC channel 6 and display analog value on the LCD**

```
//Move the potentiometer and see the values ranging from 0-4095 on 7 segment display
//
//CONNECTIONS
//Potentiometer Board
//GND GND
//VCC VCC(3.3)
//SIG GPA6

#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Seven_Segment.h"
#include "DrvGPIO.h"
#include "NUC1xx-LB_002\LCD_Driver.h"

void InitADC(void)
{
        /* Step 1. GPIO initial */
        //set the third and fourth digit below the same as adc channel number
```

```
	GPIOA->OFFD|=0x00400000;     //Disable digital input path
	SYS->GPAMFP.ADC7_SS21_AD6=1;        //Set ADC function

	/* Step 2. Enable and Select ADC clock source, and then enable ADC module */
	SYSCLK->CLKSEL1.ADC_S = 2;      //Select 22Mhz for ADC
	SYSCLK->CLKDIV.ADC_N = 1;       //ADC clock source = 22Mhz/2 =11Mhz;
	SYSCLK->APBCLK.ADC_EN = 1;    //Enable clock source
	ADC->ADCR.ADEN = 1;             //Enable ADC module

	/* Step 3. Select Operation mode */
	ADC->ADCR.DIFFEN = 0;           //single end input
	ADC->ADCR.ADMD   = 0;           //single mode

	/* Step 4. Select ADC channel */
	//0100 0000 this sets channel 6
	//basically 76543210 so whichever bit u set one,sets that adc channel
	ADC->ADCHER.CHEN = 0x40;

	/* Step 5. Enable ADC interrupt */
	ADC->ADSR.ADF =1;               //clear the A/D interrupt flags for safe
	ADC->ADCR.ADIE = 1;
	//NVIC_EnableIRQ(ADC_IRQn);

	/* Step 6. Enable WDT module */
	ADC->ADCR.ADST=1;
}

void seg_display(int value)
{
	char TEXT1[16]="Keypad:          ";
	float ans;
	int digit;
	int value1=value;
	digit= value / 1000;
	close_seven_segment();
	show_seven_segment(3,digit);
	DrvSYS_Delay(5000);

	value = value - digit * 1000;
	digit = value / 100;
	close_seven_segment();
	show_seven_segment(2,digit);
```

```
        DrvSYS_Delay(5000);

        value = value - digit* 100;
        digit = value / 10;
        close_seven_segment();
        show_seven_segment(1,digit);
        DrvSYS_Delay(5000);

        value = value - digit * 10;
        digit = value;
        close_seven_segment();
        show_seven_segment(0,digit);
        DrvSYS_Delay(5000);


  //this converts digital value shown on lcd to analog
        ans=(value1*3.3)/4095;
        Initial_panel();
        clr_all_panel();
         // scan keypad to input
        sprintf(TEXT1+8,"%f",ans); // print scankey input to string
        print_lcd(1, TEXT1);       // display string on LCD
        //DrvSYS_Delay(5000);              // delay

}

int main (void)
{
        int adc_value;

        UNLOCKREG();
        SYSCLK->PWRCON.XTL12M_EN = 1;       //Enable 12Mhz and set HCLK->12Mhz
        SYSCLK->CLKSEL0.HCLK_S = 0;
        LOCKREG();

        InitADC();

        while(1)
        {
        while(ADC->ADSR.ADF==0);       // ADC Flag, wait till 1 (A/DC conversion done)
        ADC->ADSR.ADF=1;               // write 1 to ADF is to clear the flag
        adc_value=ADC->ADDR[6].RSLT; // input 12-bit ADC value
```

```
        seg_display(adc_value);          // display value to 7-segment display

        ADC->ADCR.ADST=1;                // activate next ADC sample
                                         // 1 : conversion start
                                         // 0 : conversion stopped, ADC enter idle state
        }
}
```

**5. Program for using ADC channel 0 and display value on the 7 segment**
**//see the numbers changing on 7 seg display**

```
//CONNECTIONS
//Potentiometer Board
//GND GND
//VCC VCC(3.3)
//SIG GPA0

#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Seven_Segment.h"

void InitADC(void)
{
/* Step 1. GPIO initial */
//Should be 0x00010000 (In Q4 0x00040000)
GPIOA->OFFD |=0x00010000; //Disable digital input path
SYS->GPAMFP.ADC7_SS21_AD6=1; //Set ADC function

/* Step 2. Enable and Select ADC clock source, and then enable ADC module */
SYSCLK->CLKSEL1.ADC_S = 2; //Select 22Mhz for ADC
SYSCLK->CLKDIV.ADC_N = 1; //ADC clock source = 22Mhz/2 =11Mhz;
SYSCLK->APBCLK.ADC_EN = 1; //Enable clock source
ADC->ADCR.ADEN = 1; //Enable ADC module

/* Step 3. Select Operation mode */
ADC->ADCR.DIFFEN = 0; //single end input
ADC->ADCR.ADMD = 0; //single mode
//Should be 0x01(In Q4 0x40)
```

```
/* Step 4. Select ADC channel 0*/
ADC->ADCHER.CHEN = 0x01;

/* Step 5. Enable ADC interrupt */
ADC->ADSR.ADF =1; //clear the A/D interrupt flags for safe
ADC->ADCR.ADIE = 1;
// NVIC_EnableIRQ(ADC_IRQn);

/* Step 6. Enable WDT module */
ADC->ADCR.ADST=1;
}

void seg_display(int value)
{
int digit;
digit = value / 1000;
close_seven_segment();
show_seven_segment(3,digit);
DrvSYS_Delay(5000);
value = value - digit * 1000;

digit = value / 100;
close_seven_segment();
show_seven_segment(2,digit);
DrvSYS_Delay(5000);
value = value - digit * 100;

digit = value / 10;
close_seven_segment();
show_seven_segment(1,digit);
DrvSYS_Delay(5000);
value = value - digit * 10;
digit = value;

close_seven_segment();
show_seven_segment(0,digit);
DrvSYS_Delay(5000);
}

int main (void)
{
```

```
int adc_value;

UNLOCKREG();
SYSCLK->PWRCON.XTL12M_EN = 1; //Enable 12Mhz and set HCLK->12Mhz
SYSCLK->CLKSEL0.HCLK_S = 0;
LOCKREG();

InitADC();

while(1)
{

        while(ADC->ADSR.ADF==0); // ADC Flag, wait till 1 (A/DC conversion done)
        ADC->ADSR.ADF=1; // write 1 to ADF is to clear the flag
        //Should be 0 (In Q4 6)
        adc_value=ADC->ADDR[0].RSLT; // input 12-bit ADC value

        seg_display(adc_value); // display value to 7-segment display

        ADC->ADCR.ADST=1;     // activate next ADC sample
                              // 1 : conversion start
                              // 0 : conversion stopped, ADC enter idle state
}

}
```

**6. Program pwm1 and adc channel 6 and change the illumination of led ( use ADC and PWM) //green inbuilt led light intensity decreases or increases using the external potentiometer**

```
//CONNECTIONS
//Potentiometer Board
//GND GND
//VCC VCC(3.3)
//SIG GPA6

#include <stdio.h>
#include "NUC1xx.h"
#include "LCD_Driver.h"
#include "Driver\DrvADC.h"
#define BAUDRATE 9600

void InitADC(void)
{
```

```
	/* Step 1. GPIO initial */
	GPIOA->OFFD |=0x00400000;    //Disable digital input path
	SYS->GPAMFP.ADC7_SS21_AD6=1;                  //Set ADC function

	/* Step 2. Enable and Select ADC clock source, and then enable ADC module */
	SYSCLK->CLKSEL1.ADC_S = 2;      //Select 22Mhz for ADC
	SYSCLK->CLKDIV.ADC_N = 1;       //ADC clock source = 22Mhz/2 =11Mhz;
	SYSCLK->APBCLK.ADC_EN = 1;    //Enable clock source
	ADC->ADCR.ADEN = 1;                     //Enable ADC module

	/* Step 3. Select Operation mode */
	ADC->ADCR.DIFFEN = 0;           //single end input
	ADC->ADCR.ADMD   = 0;           //single mode

	/* Step 4. Select ADC channel 6 */
	ADC->ADCHER.CHEN = 0x40;

	/* Step 5. Enable ADC interrupt */
	ADC->ADSR.ADF =1;               //clear the A/D interrupt flags for safe
	ADC->ADCR.ADIE = 1;

	/* Step 6. Enable WDT module */
	ADC->ADCR.ADST=1;
}

void InitPWM1(void)
{
	/* Step 1. GPIO initial */
	SYS->GPAMFP.PWM1_AD14=1;

	/* Step 2. Enable and Select PWM clock source*/
	SYSCLK->APBCLK.PWM01_EN = 1;//Enable PWM clock
	SYSCLK->CLKSEL1.PWM01_S = 0;//Select 12Mhz for PWM clock source

	PWMA->PPR.CP01=11;                  //Prescaler 0~255, Setting 0 to stop output clock
	PWMA->CSR.CSR1=3;                   // PWM clock = clock source/(Prescaler + 1)/divider
	                                    //clock divider->0:/2, 1:/4, 2:/8, 3:/16, 4:/1

	/* Step 3. Select PWM Operation mode */
	//PWM0
	PWMA->PCR.CH1MOD=1;                         //0:One-shot mode, 1:Auto-load mode
```

```
                                                        //CNR and CMR will be auto-cleared
after setting CH0MOD form 0 to 1.
        PWMA->CNR1=0xFFFF;
        PWMA->CMR1=0x3FFF;

        PWMA->PCR.CH1INV=0;                     //Inverter->0:off, 1:on
        PWMA->PCR.CH1EN=1;              //PWM function->0:Disable, 1:Enable
        PWMA->POE.PWM1=1;               //Output to pin->0:Diasble, 1:Enable
}

void Delay(int count)
{
        while(count--)
        {
                //__NOP;
         }
}

int32_t main (void)
{
        //Enable 12Mhz and set HCLK->12Mhz
        char adc_value[15]="ADC Value:";

        UNLOCKREG();
        SYSCLK->PWRCON.XTL12M_EN = 1;
        SYSCLK->CLKSEL0.HCLK_S = 0;
        LOCKREG();

        InitPWM1();
        InitADC();

        Initial_panel();  //call initial pannel function
        clr_all_panel();

        /* Synch field transmission & Request Identifier Field transmission*/
        while(1)
        {
                while(ADC->ADSR.ADF==0);
                ADC->ADSR.ADF=1;
                PWMA->CMR1=ADC->ADDR[6].RSLT<<4;
                sprintf(adc_value+10,"%d",ADC->ADDR[6].RSLT);
                print_lcd(0, adc_value);
```

```
            Delay(20000);
            ADC->ADCR.ADST=1;
        }
}
```

**7. Using pwm0 change the illumination of external led connected to port A12**

//blue light intensity change automatically
//no connections

```
#include <stdio.h>
#include "NUC1xx.h"
#include "LCD_Driver.h"
#define BAUDRATE 9600

void InitPWM(void)
{
        /* Step 1. GPIO initial */
        SYS->GPAMFP.PWM0_AD13=1;

        /* Step 2. Enable and Select PWM clock source*/
        SYSCLK->APBCLK.PWM01_EN = 1;//Enable PWM clock
        SYSCLK->CLKSEL1.PWM01_S = 3;//Select 22.1184Mhz for PWM clock source

        PWMA->PPR.CP01=1;               //Prescaler 0~255, Setting 0 to stop output clock
        PWMA->CSR.CSR0=0;               // PWM clock = clock source/(Prescaler + 1)/divider

        /* Step 3. Select PWM Operation mode */
        //PWM0
        PWMA->PCR.CH0MOD=1;                     //0:One-shot mode, 1:Auto-load mode
                                //CNR and CMR will be auto-cleared after setting CH0MOD form 0 to 1.
        PWMA->CNR0=0xFFFF;
        PWMA->CMR0=0xFFFF;

        PWMA->PCR.CH0INV=0;                     //Inverter->0:off, 1:on
        PWMA->PCR.CH0EN=1;              //PWM function->0:Disable, 1:Enable
        PWMA->POE.PWM0=1;               //Output to pin->0:Diasble, 1:Enable
}

void Delay(int count)
{
        while(count--)
        {
```

```
                //__NOP;
        }
}

int32_t main (void)
{
        //Enable 12Mhz and set HCLK->12Mhz
        int val=0000;

        UNLOCKREG();
        SYSCLK->PWRCON.XTL12M_EN = 1;
        SYSCLK->CLKSEL0.HCLK_S = 0;
        LOCKREG();

        InitPWM();

        /* Synch field transmission & Request Identifier Field transmission*/

        while(1)
        {

                PWMA->CMR0=val;

                Delay(200);
                val++;

        }
}
```

**8. Using a interrupt switch on Development board and switch on/off a bulb using a relay**

**Relay connections**
**//red= vcc5**
**//green=gnd**
**//orange=signal (GPA0)**

**//Use the A0 to turn on and turn off the bulb**

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvUART.h"
```

```
#include "Driver\DrvSYS.h"

// External Interrupt Handler (INT button to trigger GPB15)
void EINT1Callback(void)
{
	DrvGPIO_ClrBit(E_GPA,0);
	DrvSYS_Delay(10);		// Delay
}

int main (void)
{
	UNLOCKREG();
	LOCKREG();
	DrvGPIO_Open(E_GPA, 0, E_IO_OUTPUT);
	// initial GPIO pin GPB11 for controlling Buzzer
	//DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);	//make it work like a button
	DrvGPIO_EnableEINT1(E_IO_BOTH_EDGE, E_MODE_EDGE, EINT1Callback);
	// configure external interrupt

 while(1)
	{
	 }

}
```

**9.Using input from the LDR switch on/off a bulb using a relay**

```
 //Vcc is vcc 5v
//gnd to gnd
//signal to GPA6
//When the light falls on the LDR the intensity increases on the LCD display (max of 4095)
//When you cover the LDR the ADC Value reduces. (min ~200)

#include <stdio.h>
#include "NUC1xx.h"
#include "DrvSYS.h"
#include "NUC1xx-LB_002\LCD_Driver.h"

void InitADC(void)
{
	/* Step 1. GPIO initial */
	GPIOA->OFFD|=0x00400000;	//Disable digital input path
```

```
    SYS->GPAMFP.ADC7_SS21_AD6=1;                //Set ADC function

    /* Step 2. Enable and Select ADC clock source, and then enable ADC module */
    SYSCLK->CLKSEL1.ADC_S = 2;      //Select 22Mhz for ADC
    SYSCLK->CLKDIV.ADC_N = 1;       //ADC clock source = 22Mhz/2 =11Mhz;
    SYSCLK->APBCLK.ADC_EN = 1;   //Enable clock source
    ADC->ADCR.ADEN = 1;                 //Enable ADC module

    /* Step 3. Select Operation mode */
    ADC->ADCR.DIFFEN = 0;               //single end input
    ADC->ADCR.ADMD   = 0;           //single mode

    /* Step 4. Select ADC channel */
    ADC->ADCHER.CHEN = 0x40;

    /* Step 5. Enable ADC interrupt */
    ADC->ADSR.ADF =1;               //clear the A/D interrupt flags for safe
    ADC->ADCR.ADIE = 1;
//  NVIC_EnableIRQ(ADC_IRQn);

    /* Step 6. Enable WDT module */
    ADC->ADCR.ADST=1;
}

/*---------------------------------------------------------------------------
  MAIN function
 ---------------------------------------------------------------------------*/
int32_t main (void)
{
    char TEXT1[16]="ADC Value:      ";
    UNLOCKREG();
    //SYSCLK->PWRCON.XTL12M_EN = 1; // enable external clock (12MHz)
    //SYSCLK->CLKSEL0.HCLK_S = 0;           // select external clock (12MHz)
    LOCKREG();

    InitADC();                          // initialize ADC

    Initial_panel();  // initialize LCD pannel
    clr_all_panel();  // clear LCD panel
    print_lcd(0, "Smpl_ADC_VR1");
```

```
        while(1)
        {
                while(ADC->ADSR.ADF==0); // wait till conversion flag = 1, conversion is done
                ADC->ADSR.ADF=1;                        // write 1 to clear the flag
                sprintf(TEXT1+10,"%4d",ADC->ADDR[6].RSLT); // convert ADC7 value into text
                print_lcd(1, TEXT1);        // output TEXT to LCD
                DrvSYS_Delay(20000);    // delay
                ADC->ADCR.ADST=1;                       // restart ADC sample
        }
}
```

**10. using stepper motor turn 180 degree forward and backward after a delay**

```
// Sampl_GPIO_StepMotor
// 5V Step Motor 28BYJ-48, driver IC = ULN2003A
// Driver board connections:
// ULN2003A NUC140
//Connections
// IN1 to GPA3
// IN2 to GPA2
// IN3 to GPA1
// IN4 to GPA0
//VCC to VCC(not 3.3 or 5)
//GND to GND

#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"
// Definitions for Step Motor turning degree

unsigned char CW[8] ={0x09,0x01,0x03,0x02,0x06,0x04,0x0c,0x08}; //Clockwise Sequence
unsigned char CCW[8]={0x08,0x0c,0x04,0x06,0x02,0x03,0x01,0x09}; //Counter-Clockwise
Sequence

void CW_MOTOR(uint16_t deg)
{
int i=0,j=0;
        for(j=0;j<(deg);j++)
        {
                for(i=0;i<8;i++)
                {
                        GPIOA->DOUT=CW[i];
```

```
                        DrvSYS_Delay(2000);             //delay 2000us = 2ms
                }
        }
}

void CCW_MOTOR(uint16_t deg)
{
int i=0,j=0;
        for(j=0;j<(deg);j++)
        {
                for(i=0;i<8;i++)
                        {
                        GPIOA->DOUT=CCW[i];
                        DrvSYS_Delay(2000);     //delay 2000us = 2ms
                        }
                }
        }

int main (void)
{
CW_MOTOR(512/2);            // Clockwise for 360 degree
//CCW_MOTOR(512/2);         // Counter-Clockwise for 180 degree
}
```